Classification using Hierarchical Naive Bayes models

Langseth, Helge; Dyhre Nielsen, Thomas

Published in:
Machine Learning

DOI (link to publication from Publisher):
10.1007/s10994-006-6136-2

Publication date:
2006

Document Version
Early version, also known as pre-print

Classification using Hierarchical Naïve Bayes models

Helge Langseth* (helgel@math.ntnu.no)
Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway

Thomas D. Nielsen (tdn@cs.auc.dk)
Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark

Abstract. Classification problems have a long history in the machine learning literature. One of the simplest, and yet most consistently well-performing set of classifiers is the Naïve Bayes models. However, an inherent problem with these classifiers is the assumption that all attributes used to describe an instance are conditionally independent given the class of that instance. When this assumption is violated (which is often the case in practice) it can reduce classification accuracy due to "information double-counting" and interaction omission.

In this paper we focus on a relatively new set of models, termed Hierarchical Naïve Bayes models. Hierarchical Naïve Bayes models extend the modeling flexibility of Naïve Bayes models by introducing latent variables to relax some of the independence statements in these models. We propose a simple algorithm for learning Hierarchical Naïve Bayes models in the context of classification. Experimental results show that the learned models can significantly improve classification accuracy as compared to other frameworks.

Keywords: Classification, Naïve Bayes models, Hierarchical models.

1. Introduction

Classification is the task of predicting the class of an instance from a set of attributes describing that instance, i.e., to apply a mapping from the attribute space into a predefined set of classes. When learning a classifier we seek to generate such a mapping based on a database of labeled instances. Classifier learning, which has been an active research field over the last decades, can therefore be seen as a model selection process where the task is to find a single model, from some set of models, with the highest classification accuracy. The set of *Naïve Bayes* (NB) models (Duda and Hart, 1973) is a family of particularly simple models which has shown to offer very good classification accuracy. NB models assume that all attributes are conditionally independent given the class. However, this assumption is clearly violated in many real-world problems; in such situations overlapping information may be given an

* Current affiliation: SINTEF Technology and Society, N-7465 Trondheim, Norway; helge.langseth@sintef.no.

unwarranted weight by the classifier (the information related to dependent attributes are treated as coming from independent sources). To resolve this problem, methods for handling the conditional dependence between the attributes have become a lively research area; these methods are typically grouped into three categories: *Feature selection* (Kohavi and John, 1997), *feature grouping* (Kononenko, 1991; Pazzani, 1996a), and *correlation modeling* (Friedman et al., 1997).

The approach taken in this paper is based on correlation modeling using Hierarchical Naïve Bayes (HNB) models (Zhang, 2004b; Zhang et al., 2003), see also Langley (1993), Spirtes et al. (1993), Martin and VanLehn (1994), and Pazzani (1996b). HNBs are tree-shaped Bayesian networks, with latent variables between the class node (the root of the tree) and the attributes (the leaves), see Figure 1. The latent variables are introduced to relax some of the independence statements of the NB classifier. For example, in the HNB model shown in Figure 1, the attributes A_1 and A_2 are not independent given C because the latent variable L_1 is unobserved. Note that if there are no latent variables in the HNB, it reduces to an NB model.

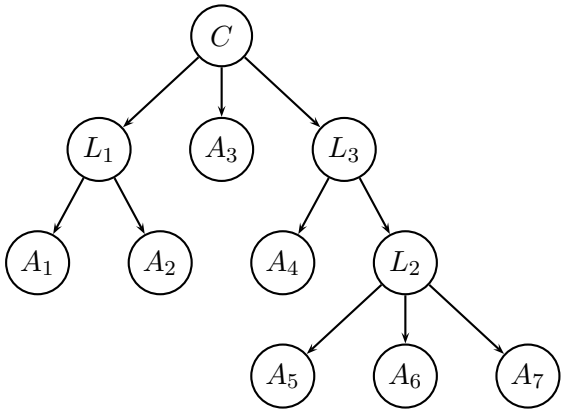


Figure 1. An HNB designed for classification. The class attribute C is the root, and the attributes $\mathcal{A} = \{A_1, \ldots, A_7\}$ are leaf nodes. L_1, L_2 and L_3 are latent variables.

The idea to use HNBs for classification was first explored by Zhang et al. (2003). However, they did not focus on classification accuracy, but rather on the generation of interesting latent structures. In particular, Zhang et al. (2003) search for the model maximizing the BIC score (a form of penalized log likelihood (Schwarz, 1978)), and such a global score function is not necessarily suitable for learning a classifier.

In this paper we focus on learning HNBs for classification: We propose a computationally efficient learning algorithm that is significantly more accurate than the system by Zhang et al. (2003) as well as several

other state-of-the-art classifiers. The proposed algorithm endows the latent variables (including their state spaces) with an explicit semantics, which may allow the decision maker to inspect the rules that governs the classification of a particular instance; informally, a latent variable can be seen as aggregating the information from its children which is relevant for classification.

The remainder of this paper is organized as follows: In Section 2 we give a brief overview of some approaches to Bayesian classification, followed by an introduction to HNB models in Section 3. In Section 4 we present an algorithm for learning HNB classifiers from data, and Section 5 is devoted to empirical results. Finally we make some concluding remarks in Section 6.

2. Bayesian classifiers

A Bayesian network (BN) (Pearl, 1988; Jensen, 2001) is a powerful tool for knowledge representation, and it provides a compact representation of a joint probability distribution over a set of variables. Formally, a BN over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ is denoted by $B = (B_S, \mathbf{\Theta}_{B_S})$, where B_S is a directed acyclic graph and $\mathbf{\Theta}_{B_S}$ is the set of conditional probabilities. To describe B_S, we let $\mathrm{pa}\,(X_i)$ denote the parents of X_i in B_S, and $\mathrm{ch}\,(X_i)$ is the children of X_i in B_S. We use $\mathrm{sp}\,(X_i)$ to denote the state-space of X_i (i.e., the set of values the variable X_i can take), and for a set of variables we have $\mathrm{sp}\,(\mathcal{X}) = \times_{X \in \mathcal{X}} \mathrm{sp}\,(X)$. We use the notation $X_i \perp\!\!\!\perp X_j$ to denote that X_i is independent of X_j and $X_i \perp\!\!\!\perp X_j \,|\, X_k$ for conditional independence of X_i and X_j given X_k. In the context of classification, we shall use C to denote the class variable ($\mathrm{sp}\,(C)$ is the set of possible classes), and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is the set of attributes describing the possible instances to be classified.

When doing classification in a probabilistic framework, a new instance (described by $\boldsymbol{a} \in \mathrm{sp}\,(\mathcal{A})$) is classified to class c^* according to:

$$c^* = \arg \min_{c \in \mathrm{sp}(C)} \sum_{c' \in \mathrm{sp}(C)} L(c|c') P(C = c' \,|\, \boldsymbol{A} = \boldsymbol{a}),$$

where $L(\cdot|\cdot)$ defines the *loss function*, i.e., $L(c|c')$ is the cost of classifying an instance to class c when the correct class is c'. The most commonly used loss function is the 0/1-loss, defined s.t. $L(c|c') = 0$ if $c' = c$ and 1 otherwise. When using the 0/1-loss function, the expression above can be rewritten as:

$$c^* = \arg \max_{c \in \mathrm{sp}(C)} P(C = c \,|\, \boldsymbol{A} = \boldsymbol{a}),$$

which implies that an instance $\boldsymbol{a}$ is classified to the most probable class c^* given that instance.

Since we rarely have access to $P(C = c \,|\, \boldsymbol{A} = \boldsymbol{a})$, learning a classifier amounts to estimating this probability distribution from a set of labeled training samples which we denote by $\mathcal{D}_N = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_N\}$; N is the number of training instances and $\boldsymbol{D}_i = \left(c^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)}\right)$ is the class and attributes of instance i, $i = 1, \ldots, N$. Let $P(C = c \,|\, \boldsymbol{A} = \boldsymbol{a}, \mathcal{D}_N)$ be the *a posteriori* conditional probability for $C = c$ given $\boldsymbol{A} = \boldsymbol{a}$ after observing $\mathcal{D}_N$. Then an optimal Bayes classifier will classify a new instance with attributes $\boldsymbol{a}$ to class c^* according to (see, e.g., Mitchell (1997)):

$$c^* = \arg \min_{c \in \mathrm{sp}(C)} \sum_{c' \in \mathrm{sp}(C)} L(c, c') P(C = c' \,|\, \boldsymbol{a}, \mathcal{D}_N).$$

An immediate approach to calculate $P(C = c \,|\, \boldsymbol{A} = \boldsymbol{a}, \mathcal{D}_N)$ is to use a standard BN learning algorithm, where the training data is used to give each possible classifier a *score* which signals its appropriateness as a classification model. One such scoring function is based on the minimum description length (MDL) principle (Rissanen, 1978; Lam and Bacchus, 1994):

$$\mathrm{MDL}(B \,|\, \mathcal{D}_N) = \frac{\log N}{2} \left|\widehat{\boldsymbol{\Theta}}_{B_S}\right| - \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)}, \boldsymbol{a}^{(i)} \,\middle|\, \widehat{\boldsymbol{\Theta}}_{B_S}\right)\right).$$

That is, the best scoring model is the one that minimizes $\mathrm{MDL}(\cdot \,|\, \mathcal{D}_N)$, where $\widehat{\boldsymbol{\Theta}}_{B_S}$ is the maximum likelihood estimate of the parameters in the model, and $\left|\widehat{\boldsymbol{\Theta}}_{B_S}\right|$ is the dimension of the parameter space (i.e., the number of free parameters in the model). However, as pointed out by Greiner et al. (1997) and Friedman et al. (1997) a "global" criterion like MDL may not be well suited for learning a classifier, as:

$$\begin{aligned}
&\sum_{i=1}^{N} \log\left(P_B\left(c^{(i)}, \boldsymbol{a}^{(i)}, \widehat{\boldsymbol{\Theta}}_{B_S}\right)\right) = \\
&\quad \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)} \,\middle|\, \boldsymbol{a}^{(i)}, \widehat{\boldsymbol{\Theta}}_{B_S}\right)\right) + \sum_{i=1}^{N} \log\left(P_B\left(a_1^{(i)}, \ldots, a_n^{(i)}, \widehat{\boldsymbol{\Theta}}_{B_S}\right)\right).
\end{aligned}$$

In the equation above, the first term on the right-hand side measures how well the classifier performs on $\mathcal{D}_N$, whereas the second term measures how well the classifier estimates the joint distribution over the attributes. Thus, only the first term is related to the classification task, and the latter term will therefore merely bias the model search; in fact,

the latter term will dominate the score if n is large. To overcome this problem, Friedman et al. (1997) propose to replace MDL with *predictive* MDL, MDL_p, defined as:

$$\mathrm{MDL}_p(B \,|\, \mathcal{D}_N) = \frac{\log N}{2} \left|\widehat{\boldsymbol{\Theta}}_{B_S}\right| - \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)} \,\middle|\, \boldsymbol{a}^{(i)}\,,\, \widehat{\boldsymbol{\Theta}}_{B_S}\right)\right). \quad (1)$$

However, as also noted by Friedman et al. (1997), MDL_p cannot be calculated efficiently in general.

The argument leading to the use of MDL_p as a scoring function rests upon the asymptotic theory of statistics. That is, model search based on MDL_p is guaranteed to select the best classifier w.r.t. 0/1-loss when $N \to \infty$. Unfortunately, though, the score may not be successful for finite data sets (Friedman, 1997). To overcome this potential drawback, Kohavi and John (1997) describe the *wrapper approach*. Informally, this method amounts to estimating the accuracy of a given classifier by cross validation (based on the *training data*), and to use this estimate as the scoring function. The wrapper approach relieves the scoring function from being based on approximations of the classifier design, but at the potential cost of higher computational complexity. In order to reduce this complexity when learning a classifier, one approach is to focus on a particular sub-class of BNs. Usually, these sub-classes are defined by the set of independence statements they encode. For instance, one such restricted set of BNs is the Naïve Bayes models which assume that $A_i \perp\!\!\!\perp A_j \,|\, C$, i.e., that $P(C = c|\boldsymbol{A} = \boldsymbol{a}) \propto P(C = c) \prod_{i=1}^{n} P(A_i = a_i|C = c)$.

Even though the independence statements of the NB models are often violated in practice, these models have shown to provide surprisingly good classification results; the ranking of the classes may be correct even though the posterior conditional probability for the class variable is inaccurate. For instance, Zhang (2004a) shows that conditional dependences among the attributes do not affect classification accuracy as long as the dependences distribute evenly among the classes or if they cancel out. Other research into explaining the merits of the NB model has emphasized the difference between the 0/1-loss function and the log-loss, see e.g. Friedman (1997) and Domingos and Pazzani (1997). Friedman (p. 76, 1997) concludes:

> Good probability estimates are not necessary for good classification; similarly, low classification error does not imply that the corresponding class probabilities are being estimated (even remotely) accurately.

The starting point of Friedman (1997) is that a classifier learned for a particular domain is a function of the training set. As the training set is considered a random sample from the domain, the classifier generated by a learner can be seen as a random variable; we shall use $\hat{P}(C = c \mid \boldsymbol{A})$ to denote the learned classifier. Friedman (1997) characterizes a (binary) classifier by its bias (i.e., $\mathbb{E}_{\mathcal{D}_N}\left[P(C \mid \boldsymbol{A}) - \hat{P}(C \mid \boldsymbol{A})\right]$) and its variance (i.e., $\mathrm{Var}_{\mathcal{D}_N}\left(\hat{P}(C \mid \boldsymbol{A})\right)$); the expectations are taken over all possible training sets of size N. Friedman (1997) shows that in order to learn classifiers with low 0/1-loss it may not be sufficient to simply focus on finding a model with negligible classifier bias; robustness in terms of low classifier variance can be just as important.

An example of a class of models where negligible asymptotic bias (i.e., fairly high model expressibility) is combined with robustness is the *Tree Augmented Naïve Bayes* (TAN) models, see Friedman et al. (1997). TAN models relax the NB assumption by allowing a more general correlation structure between the attributes. More specifically, a Bayesian network model is initially created over the variables in $\mathcal{A}$, and this model is designed s.t. each variable A_i has at most one parent (that is, the structure is a *directed tree*). Afterwards, the class attribute is included in the model by making it the *parent* of each attribute. Friedman et al. (1997) use an adapted version of the algorithm by Chow and Liu (1968) to learn the classifier, and they prove that the structure they find is the TAN model which maximizes the likelihood of $\mathcal{D}_N$; the algorithm has time complexity $O\left(n^2(N + \log(n))\right)$. The TAN model has clearly more expressive power than the NB model (Jaeger, 2003), however, it can still not represent all correlation structures among the attributes. As an alternative we consider another class of models (called Hierarchical Naïve Bayes models) that can in principle model any such correlation structure.

3. Hierarchical Naïve Bayes models

A special class of Bayesian networks is the so-called Hierarchical Naïve Bayes (HNB) models (Zhang et al., 2003), see also Kočka and Zhang (2002), and Zhang (2004b). An HNB is a tree-shaped Bayesian network with only discrete variables, and where the variables are partitioned into three disjoint sets: $\{C\}$ is the class variable, $\mathcal{A}$ is the set of attributes, and $\mathcal{L}$ is a set of *latent* (or *hidden*) variables. In the following we use A to represent an attribute, whereas L is used to denote a latent variable; X and Y denote variables that may be either attributes or latent variables. In an HNB the class variable C is the root of the tree

($\mathrm{pa}(C) = \emptyset$) and the attributes are at the leaves ($\mathrm{ch}(A) = \emptyset, \forall A \in \mathcal{A}$); the latent variables are all internal ($\mathrm{ch}\,(L) \neq \emptyset$, $\mathrm{pa}\,(L) \neq \emptyset$, $\forall\, L \in \mathcal{L}$). The use of latent variables allows conditional dependencies to be encoded in the model (as compared to, e.g., the NB model). For instance, by introducing a latent variable as a parent of the attributes A_i and A_j, we can represent the (local) dependence statement $A_i \not\perp\!\!\!\perp A_j | C$ (see for instance A_1 and A_2 in Figure 1). Being able to model such local dependencies is particularly important for classification, as overlapping information would otherwise be double-counted:

Example 1. Consider a domain consisting of two classes ($C = 0$ or $C = 1$), and two binary attributes A_1 and A_2. Assume that A_1 and A_2 always have the same value, i.e., $P(A_1 = A_2) = 1$, and let $P(A_i = k \,|\, C = k) = 3/5$, for $i = 1, 2$, $k = 0, 1$, and $P(C = 0) = 2/3$. Consequently, we have:

$$\begin{aligned} P(C = 0 \,|\, \boldsymbol{A} = \mathbf{1}) &= \frac{P(A_1 = 1, A_2 = 1 \,|\, C = 0) P(C = 0)}{P(A_1 = 1, A_2 = 1)} \\ &= \frac{2/5 \cdot 2/3}{2/5 \cdot 2/3 + 3/5 \cdot 1/3} = \frac{4}{7} \end{aligned}$$

and therefore $P(C = 0 \,|\, A_1 = 1, A_2 = 1) > P(C = 1 \,|\, A_1 = 1, A_2 = 1)$. On the other hand, if we were to encode this domain in a Naïve Bayes structure, we would get:

$$\begin{aligned} P(C = 0 \,|\, \boldsymbol{A} = \mathbf{1}) &= \frac{P(A_1 = 1, A_2 = 1 \,|\, C = 0) P(C = 0)}{\sum_j P(A_1 = 1, A_2 = 1 \,|\, C = j) \cdot P(C = j)} \\ &= \frac{P(A_1 = 1 \,|\, C = 0) P(A_2 = 1 \,|\, C = 0) P(C = 0)}{\sum_j P(A_1 = 1 | C = j) P(A_2 = 1 | C = j) P(C = j)} \\ &= \frac{2/5 \cdot 2/5 \cdot 2/3}{2/5 \cdot 2/5 \cdot 2/3 + 3/5 \cdot 3/5 \cdot 1/3} = 8/17, \end{aligned}$$

hence $P(C = 0 \,|\, A_1 = 1, A_2 = 1) < P(C = 1 \,|\, A_1 = 1, A_2 = 1)$, which would revert the classification if 0/1-loss is used.

Note that the HNB model reduces to the NB model in the special case when there are no latent variables.

When learning an HNB we can restrict our attention to the *parsimonious* HNB models; we need not consider models which encode a probability distribution that is also encoded by another model with fewer parameters. Formally, an HNB model, $M = (B_S, \Theta_{B_S})$, with class variable C and attribute variables $\mathcal{A}$ is said to be parsimonious if there does not exist another HNB model, $M' = (B'_S, \Theta'_{B_S})$, with the same class and attribute variables s.t.:

i) M' has fewer parameters than M, i.e., $|\Theta_{B_S}| > |\Theta'_{B_S}|$.

ii) The probability distributions over the class and attribute variables are the same in the two models, i.e., $P_M(C, \boldsymbol{A}) = P_{M'}(C, \boldsymbol{A})$.

In order to obtain an operational characterization of these models, Zhang et al. (2003) define the class of *regular* HNB models. An HNB model is said to be regular if for any latent variable L, with neighbors (parent and children) $X_1, X_2, \ldots, X_n$, it holds that:

$$|\mathrm{sp}(L)| \leq \frac{\prod_{i=1}^{n} |\mathrm{sp}(X_i)|}{\max_{i=1,\ldots,n} |\mathrm{sp}(X_i)|}. \tag{2}$$

Strict inequality must hold whenever L has only two neighbors and at least one of them is a latent node.[1]

Zhang et al. (2003) show that *i*) any parsimonious HNB model is regular, and *ii*) for a given set of class and attribute variables, the set of regular HNB model structures is finite. Observe that these two properties ensure that when searching for an HNB model we only need to consider regular HNB models and we need not deal with infinite search spaces.[2]

As opposed to other frameworks, such as NB or TAN models, an HNB can model any correlation among the attribute variables given the class by simply choosing the state-spaces of the latent variables large enough (although this encoding is not necessarily done in a cost-effective manner in terms of model complexity); note that the independence statements are not always represented explicitly in the graphical structure, but are sometimes only encoded in the conditional probability tables. On the other hand, the TAN model, for instance, is particularly efficient for encoding such statements but may fail to represent certain types of dependence relations among the attribute variables.

Example 2. Consider the classification rule "$C = 1$ if and only if exactly two out of the three binary attributes A_1, A_2 and A_3 are in state 1". Obviously, a Naïve Bayes model can not represent this statement, and neither can the TAN model; see Jaeger (2003) for a discussion regarding the expressibility of probabilistic classifiers.

On the other hand, an HNB can (somewhat expensively) do it as follows: As the only child of the binary class variable we introduce the latent variable L, with $\mathrm{sp}(L) = \{0, 1, \ldots, 7\}$ and $\mathrm{ch}(L) = \{A_1, A_2, A_3\}$. A_j $(j = 1, 2, 3)$ is given deterministically by its parent, and its CPT is defined s.t. $A_j = 1$ iff bit j equals 1 when the value of L is given in binary representation. If, for instance, $L = 5$ (`101` in binary representation), then $A_1 = 1$, $A_2 = 0$, and $A_3 = 1$ (with probability 1). Thus, all information about the attributes are contained in L, and we

can simply use the classification rule $C = 1$ iff $L \in \{3, 5, 6\}$, which can be encoded by insisting that $P(L = l|C = 0)$ is 0 for $l \in \{3, 5, 6\}$ and strictly positive otherwise, whereas $P(L = l|C = 1)$ is strictly positive iff $l \in \{3, 5, 6\}$ and 0 otherwise.

More generally, by following the method above we see that any conditional correlation among the attributes can, in principle, be modeled by an HNB: Simply introduce a single latent variable L having all the attributes as children and with the state space defined as $\mathrm{sp}(L) = \times_{i=1}^{n} \mathrm{sp}(A_i)$. Clearly, this structure can encode any conditional distribution over the attributes.

4. Learning HNB classifiers

Learning an HNB model for classification has previously been explored by Zhang et al. (2003), with the aim of finding a scientific model with an interesting latent structure. Their method is based on a hill-climbing algorithm where the models are scored using the BIC score. However, a drawback of the algorithm is its high computational complexity, and, as discussed in Section 2, the type of scoring function being used does not necessarily facilitate the identification of an accurate classifier. In particular, Zhang et al. (p. 284, 2003) state that

> [...] the primary goal of this paper is to discover interesting latent variables rather than to improve classification accuracy.

In what follows we take a different approach as we focus on learning HNB models with the sole aim of obtaining an accurate classifier.[3] We also demonstrate the feasibility of the algorithm in terms of its computational complexity, and we describe an inference procedure which is tailored for the learned models.

4.1. The main algorithm

Our learning algorithm is based on a greedy search over the space of HNBs; we initiate the search with an HNB model, H_0, and learn a sequence $\{H_k\}$, $k = 0, 1, \ldots$ of HNB models. The search is conducted s.t. at each step k we investigate the *search boundary* (denoted $\mathcal{B}(H_k)$) of the current model H_k, i.e., the set of models that can be reached from H_k in a single step. We then score all models $H \in \mathcal{B}(H_k)$ and pick the best scoring one. This is repeated until no higher scoring model can be found.

The search boundary is defined s.t. the HNB structure is grown incrementally starting from the NB model. More specifically, if $\mathcal{L}_k$ is

the set of latent variables in model H_k, then the set of latent variables in H_{k+1}, is enlarged s.t. $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{L\}$, where L is a new latent variable. We restrict ourselves to only consider candidate latent variables which are parents of two variables X and Y where $\{X, Y\} \subseteq \mathrm{ch}(C)$ in H_k.[4] Hence, we define H_{k+1} as the HNB which is produced from H_k by including a latent variable L s.t. $\mathrm{pa}(L) = \{C\}$ and $\mathrm{pa}(X) = \mathrm{pa}(Y) = \{L\}$; H_{k+1} is otherwise structurally identical to H_k. Thus, the search boundary $\mathcal{B}(H_k)$ consists of all models where exactly one latent variable has been added to H_k; there is one such model in $\mathcal{B}(H_k)$ for each possible definition of the state-space for the new latent variable. Note that since we use the NB model structure as our starting point (H_0) each H_k is a tree with a binary internal structure, i.e., any latent node $L' \in \mathcal{L}_k$ has exactly two children but the class node C may have up to n children.

Unfortunately, since $\mathcal{B}(H_k)$ contains a model for each possible specification of the state space of each possible latent variable it is not computationally feasible to evaluate all the models in the search boundary. To overcome this problem we instead select $\kappa > 1$ models (contained in $\mathcal{B}(H_k)$) to represent the search boundary, and then we evaluate these models. Specifically, when identifying a particular model in this set we proceed in two steps:

1) decide where to insert the latent variable;

2) define the state-space of the latent variable.

In order to increase the robustness of the algorithm (e.g., in case of outliers), we pick the $\kappa > 1$ models as follows: Randomly partition the training data $\mathcal{D}_N$ into κ partly overlapping subsets, each containing $100(\kappa - 1)/\kappa\%$ of the training data, and then use each of these subsets to approximate the *best* model in the search boundary (following the two-step procedure above). This results in a list of up to κ different candidate models which are used to represent $\mathcal{B}(H_k)$. Note that when using the above two-step procedure for identifying a latent variable, we cannot use scoring functions such as the wrapper approach, MDL, or MDL_p in the *first* step since this step does not select a completely specified HNB.

From the set of models representing the search boundary we then select a model with a higher score than the current one; if no such model can be found, then the current model is returned. The score-function is defined s.t. a high value corresponds to what is thought to be a model with good classification qualities (as measured by the average loss on unseen data), i.e., $\mathrm{Score}(H \mid \mathcal{D}_N)$ measures the "goodness" of H. In order to apply a score metric that is closely related to what the search

algorithm tries to achieve, we use the wrapper approach by Kohavi and John (1997). That is, we use cross validation (over the training set $\mathcal{D}_N$) to estimate an HNB's classification accuracy on unseen data; notice that the test-set (if specified) is *not* used by the learner. To summarize, the structure of the algorithm can be outlined as follows:

Algorithm 1. (Skeleton)

1. Initiate model search with H_0.
2. Partition the training-set into κ subsets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(\kappa)}$.
3. **For** $k = 0, 1, \ldots, n-1$:[5]
   a) **For** $i = 1, \ldots, \kappa$:
      i) Select a candidate latent variable $L^{(i)}$ according to $\mathcal{D}^{(i)}$.
      ii) Select the state-space of $L^{(i)}$.
      iii) Define $H^{(i)}$ by "including" $L^{(i)}$ in H_k.
   b) $H' = \arg \max_{i=1,\ldots,\kappa} \text{Score}\left(H^{(i)} \,|\, \mathcal{D}_N\right)$.
   c) **If** $\text{Score}(H' \,|\, \mathcal{D}_N) > \text{Score}(H_k \,|\, \mathcal{D}_N)$ **then**:
      $H_{k+1} \leftarrow H'$; $k \leftarrow k+1$.
      else
      return H_k.
4. **Return** H_n.

Before describing Step (i) and Step (ii) in detail, we recall that the algorithm starts out with an NB model, and that the goal is to introduce latent variables to improve upon that structure, i.e., to avoid "double-counting" of information when the independence statements of the NB model are violated.

4.1.1. *Step (i): Finding a candidate latent variable*
To facilitate the goal of the algorithm, a latent variable L is proposed as the parent of $\{X, Y\} \subseteq \text{ch}(C)$ if the data points towards $X \not\perp\!\!\!\perp Y \,|\, C$. That is, we consider variables that are strongly correlated given the class variable as indicating a promising position for including a latent variable; from this perspective there is no reason to introduce a latent variable as a parent of X and Y if $X \perp\!\!\!\perp Y \,|\, C$. Hence, the variables that have the highest correlation given the class variable may be regarded as the most promising candidate-pair. More specifically, we calculate the conditional mutual information given the class variable, $I(\cdot, \cdot \,|\, C)$, for all (unordered) pairs $\{X, Y\} \subseteq \text{ch}(C)$.[6] However, as $I(X, Y \,|\, C)$ is

increasing in both $|\mathrm{sp}(X)|$ and $|\mathrm{sp}(Y)|$ we cannot simply pick the pair $\{X, Y\}$ that maximizes $I(X, Y \mid C)$; this strategy would unintentionally bias the search towards latent variables with children having large state-spaces. Instead we utilize that under the assumption that $X \perp\!\!\!\perp Y \mid C$ we have:

$$2N \cdot I(X, Y \mid C) \xrightarrow{\mathcal{L}} \chi^2_\nu,$$

where $\nu = |\mathrm{sp}(C)|\,(|\mathrm{sp}(X)| - 1)\,(|\mathrm{sp}(Y)| - 1)$, and $\xrightarrow{\mathcal{L}}$ means convergence in distribution as $N \to \infty$, see, e.g., Whittaker (1990). Finally, we let $i(X, Y)$ be the estimated value of $I(X, Y \mid C)$, and calculate

$$Q(X, Y \mid \mathcal{D}_N) = P\left(Z \geq 2N \cdot i(X, Y)\right), \tag{3}$$

where Z is χ^2 distributed with *df* degrees of freedom, that is, $Q(X, Y \mid \mathcal{D}_N)$ gives the p-value of a hypothesis test of H_0: $X \perp\!\!\!\perp Y \mid C$. The pairs $\{X, Y\}$ are ordered according to these probabilities, s.t. the pair with the lowest probability is picked out. By selecting the pairs of variables according to $Q(X, Y \mid \mathcal{D}_N)$, the correlations are normalized w.r.t. the size differences in the state-spaces.

Unfortunately, to greedily select a pair of highly correlated variables as the children of a new latent variable is not always the same as improving classification accuracy, as can be seen from the example below.

Example 3. Consider a classifier with binary attributes $\mathcal{A} = \{A_1, A_2, A_3\}$ (all with uniform marginal distributions) and target concept $C = 1 \Leftrightarrow \{A_1 = 1 \wedge A_2 = 1\}$. Assume that A_1 and A_2 are marginally independent but that $P(A_2 = A_3) = 0.99$. It then follows that:

$$P\left(Q(A_2, A_3 \mid \mathcal{D}_N) < Q(A_1, A_2 \mid \mathcal{D}_N)\right) \to 1$$

as N grows large (the uncertainty is due to the random nature of $\mathcal{D}_N$). Hence, the heuristic will not pick out $\{A_1, A_2\}$ which in a myopic sense appears to be most beneficial w.r.t. classification accuracy, but will propose to add a variable L' with children $\mathrm{ch}(L') = \{A_2, A_3\}$. Luckily, as we shall see in Example 4, this does not necessarily affect classification accuracy.

4.1.2. *Step (ii): Selecting the state-space*

To find the cardinality of a latent variable L, we use an algorithm similar to the one by Elidan and Friedman (2001): Initially, the latent variable is defined s.t. $|\mathrm{sp}(L)| = \prod_{X \in \mathrm{ch}(L)} |\mathrm{sp}(X)|$, where each state of L corresponds to exactly one combination of the states of the children of L. Let the states of the latent variable be labeled $l_1, \ldots, l_t$. We then

iteratively collapse two states l_i and l_j into a single state l^* as long as this is "beneficial". This approach implies that a latent variable can be seen as aggregating the information from its children which is relevant for classification. Moreover, as we shall see later in this section, this semantic interpretation allows us to *infer* data for the latent variables due to the deterministic relations encoded in the model.

Now, ideally we would measure the benefit of collapsing two states using the wrapper approach, but as this is computationally expensive we shall instead use the MDL_p score to approximate the classification accuracy. Let $H' = (B'_S, \boldsymbol{\Theta}_{B'_S})$ be the HNB model obtained from a model $H = (B_S, \boldsymbol{\Theta}_{B_S})$ by collapsing states l_i and l_j. Then l_i and l_j should be collapsed if and only if $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) = \mathrm{MDL}_p\,(H \,|\, \mathcal{D}_N) - \mathrm{MDL}_p\,(H' \,|\, \mathcal{D}_N) > 0$. For each pair (l_i, l_j) of states we therefore compute:

$$\begin{aligned}
\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) &= \mathrm{MDL}_p(H|\mathcal{D}_N) - \mathrm{MDL}_p(H'|\mathcal{D}_N) \\
&= \frac{\log(N)}{2}\left(|\boldsymbol{\Theta}_{B_S}| - \left|\boldsymbol{\Theta}_{B'_S}\right|\right) \\
&\quad + \sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right) - \log\left(P_H(c^{(i)}|a^{(i)})\right)\right].
\end{aligned}$$

For the second term we first note that:

$$\begin{aligned}
&\sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right) - \log\left(P_H(c^{(i)}|a^{(i)})\right)\right] \\
&\qquad = \sum_{i=1}^{N}\log\frac{P_{H'}(c^{(i)}|a^{(i)})}{P_H(c^{(i)}|a^{(i)})} \\
&\qquad = \sum_{D\in\mathcal{D}_N : f(D,l_i,l_j)}\log\frac{P_{H'}\left(c^D|a^D\right)}{P_H\left(c^D|a^D\right)},
\end{aligned}$$

where $f(D, l_i, l_j)$ is true if case D includes either $\{L = l_i\}$ or $\{L = l_j\}$; cases that do not include these states cancel out. This is also referred to as *local decomposability* by Elidan and Friedman (2001), i.e., the gain of collapsing two states l_i and l_j is local to those states and it does not depend on whether or not other states have been collapsed. Note that in order to calculate $f(\cdot)$ we exploit that the deterministic relations between the latent variables and their children allows us to infer actual values for the latent variables for any configuration over the attributes (we shall return to this issue later in this section).

In order to avoid considering all possible combinations of the attributes we approximate the difference in predictive MDL as the difference w.r.t. the relevant subtree. The relevant subtree is defined by

C together with the subtree having L as root:[7]

$$\sum_{D\in\mathcal{D}_N:f(D,l_i,l_j)} \log \frac{P_{H'}\left(c^D|a^D\right)}{P_H\left(c^D|a^D\right)} \approx \log \prod_{c\in \mathrm{sp}(C)} \left[\frac{\left(\frac{N(c,l_i)+N(c,l_j)}{N(l_i)+N(l_j)}\right)^{N(c,l_i)+N(c,l_j)}}{\left(\frac{N(c,l_i)}{N(l_i)}\right)^{N(c,l_i)} \cdot \left(\frac{N(c,l_j)}{N(l_j)}\right)^{N(c,l_j)}} \right], \quad (4)$$

where $N(c,s)$ and $N(s)$ are the sufficient statistics. I.e., $N(c,s)$ is the number of cases in the database where $C = c$ *and* $L = s$, and $N(s) = \sum_{c\in\mathrm{sp}(C)} N(c,s)$ is the number of cases where $L = s$. We shall return to the accuracy of the approximation later in this section.

States are collapsed in a greedy manner, i.e., we find the pair of states with highest $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N)$ and collapse those two states if $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) > 0$. This is repeated (making use of local decomposability) until no states can be collapsed:

Algorithm 2. (Determine state-space of L)

1. Initiate state-space s.t. $|\mathrm{sp}(L)| = \prod_{X\in\mathrm{ch}(L)} |\mathrm{sp}(X)|$.
   Label the states s.t. each state corresponds to a unique combination of $\mathrm{ch}(L)$.
2. **For each** $l_i, l_j \in \mathrm{sp}(L)$ **do**:
   Calculate $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N)$.
3. Select $l'_i, l'_j \in \mathrm{sp}(L)$ s.t. $\Delta_L(l'_i, l'_j \,|\, \mathcal{D}_N)$ is maximized.
4. **If** $\Delta_L(l'_i, l'_j \,|\, \mathcal{D}_N) > 0$ **then**:
   Collapse states l'_i and l'_j; **goto** 2.
5. **Return** state-space of L.

It should be noted that Elidan and Friedman (2001) initialize their search with one state in L for each combination of the variables in the Markov blanket of L. However, since we are only interested in regular HNB models it is sufficient to consider the smaller set of variables defined by $\mathrm{ch}(L)$ (cf. Equation 2). Actually, even with this set of variables we may still produce irregular HNB models, but as we use the difference in predictive MDL to guide the refinement of the state space we are guaranteed to arrive at a regular HNB as $N \to \infty$.[8]

Example 4. (Example 3 cont'd) The state-space of L' with $\mathrm{ch}(L') = \{A_2, A_3\}$ is collapsed by Algorithm 2 after L' is introduced. For large N

the penalty term in MDL_p ensures that the state-space will be collapsed to two states mirroring the states of A_2 because L' will not significantly change the predictive likelihood from what the model previously held (note that $P(C = c \,|\, A_2, A_3, \mathcal{D}_N) \approx P(C = c \,|\, A_2, \mathcal{D}_N)$). Hence, by introducing L' we get a more robust classifier, where the classification noise introduced by A_3 is removed; in this sense, the algorithm can be seen as incorporating a form of feature selection (Langley, 1994). The latent variable L'' with children $\mathrm{ch}\,(L'') = \{L', A_1\}$ will be introduced in the next iteration of Algorithm 1, and the target concept can eventually be learned.

Note that Algorithm 2 will only visit a subset of the HNB-models in the search boundary, namely those where the latent variables are given (deterministically) by the value of their children. An important side-effect of this is that we can give a semantic interpretation to the state-spaces of the latent variables in the models the algorithm generates: $L \in \mathcal{L}$ *aggregates* the information from its children which is relevant for classification. If, for example, L is the parent of two binary variables A_1 and A_2, then Algorithm 2 is initiated s.t. L's state-space is $\mathrm{sp}\,(L) = \{A_1 = 0 \wedge A_2 = 0, A_1 = 0 \wedge A_2 = 1, A_1 = 1 \wedge A_2 = 0, A_1 = 1 \wedge A_2 = 1\}$. When the algorithm collapses states, we can still maintain an explicit semantics over the state-space, e.g., if the first and second state is collapsed we obtain a new state defined as $(A_1 = 0 \wedge A_2 = 0) \vee (A_1 = 0 \wedge A_2 = 1)$, i.e., $A_1 = 0$.[9] Observe that this interpretation also implies that if the attributes have been produced by discretizing a collection of continuous variables (see, e.g., Fayyad and Irani (1993)), then the latent variables can be seen as encoding a form of hierarchical discretization of the original variables.

An important aspect of the semantic interpretation, is that it allows us to *infer* data for the latent variables due to the deterministic relations encoded in the model. This fact provides us with a fast calculation scheme, as we "observe" all the variables in $\mathcal{A}$ *and* $\mathcal{L}$. It therefore also follows that we can represent the HNB classifier using only the class variable and its children. Hence, the representation we will utilize is a Naïve Bayes structure where the "attributes" are represented by the variables which occur as children of the class variable in the HNB model. It is simple to realize that the number of free parameters required to represent this structure equals:

$$|\boldsymbol{\Theta}_{B_S}| = (|\mathrm{sp}\,(C)| - 1) + |\mathrm{sp}\,(C)| \sum_{X \in \mathrm{ch}\,(C)} (|\mathrm{sp}\,(X)| - 1)\,,$$

see also Kočka and Zhang (2002). Hence, the difference in predictive MDL (used in Algorithm 2) can be approximated by:

$$\begin{aligned}\Delta_L(l_i, l_j) \;\approx\; & \log_2(N)\frac{|\mathrm{sp}\,(C)|}{2} \\ & - \sum_{c\in \mathrm{sp}(C)} N(c, l_i)\log_2\left(\frac{N(c, l_i)}{N(c, l_i) + N(c, l_j)}\right) \\ & - \sum_{c\in \mathrm{sp}(C)} N(c, l_j)\log_2\left(\frac{N(c, l_j)}{N(c, l_i) + N(c, l_j)}\right) \\ & + \; N(l_i)\log\left(\frac{N(l_i)}{N(l_i) + N(l_j)}\right) + N(l_j)\log\left(\frac{N(l_j)}{N(l_i) + N(l_j)}\right).\end{aligned} \tag{5}$$

It can be shown that the approximation of Equation (5) is exact if $P\,(\mathrm{ch}\,(C) \setminus \{L\} \,|\, L = l_i) = P\,(\mathrm{ch}\,(C) \setminus \{L\} \,|\, L = l_j)$.

Finally, to summarize the steps discussed above (and formalize Step (i) and Step (ii) of Algorithm 1) we have the following algorithm:

Algorithm 3. (Learn HNB classifier)

1. Initiate model search with H_0.
2. Partition the training-set into κ partly overlapping subsets $\mathcal{D}^{(1)}, \ldots,$ $\mathcal{D}^{(\kappa)}$.
3. **For** $k = 0, 1, \ldots, n-1$:
   a) **For** $i = 1, \ldots, \kappa$:
      i) Let $\{X^{(i)}, Y^{(i)}\} = \arg\min_{\{X,Y\}\subseteq \mathrm{ch}\,(C)} Q\left(X, Y \,|\, \mathcal{D}^{(i)}\right)$ (i.e., $\left\{X^{(i)}, Y^{(i)}\right\} \subseteq \mathrm{ch}\,(C)$ in H_k), and define the latent variable $L^{(i)}$ with children $\mathrm{ch}\left(L^{(i)}\right) = \left\{X^{(i)}, Y^{(i)}\right\}$.
      ii) Collapse the state-space of $L^{(i)}$ (Algorithm 2 with $\mathcal{D}^{(i)}$ used in place of $\mathcal{D}_N$).
      iii) Define $H^{(i)}$ by introducing $L^{(i)}$ into H_k.
   b) $H' = \arg\max_{i=1,\ldots,\kappa} \mathrm{Score}\left(H^{(i)} \,|\, \mathcal{D}_N\right)$.
   c) **If** $\mathrm{Score}(H' \,|\, \mathcal{D}_N) > \mathrm{Score}(H_k \,|\, \mathcal{D}_N)$ **then**:
      $H_{k+1} \leftarrow H'$; $k \leftarrow k+1$
      else
      return H_k.
4. **Return** H_n.

It is obvious that any conditional distribution $P(A_1, \ldots, A_N \mid C)$ is in principle reachable by the search algorithm but, as the score function is multi-modal over the search space, the search will in general only converge towards a local optimum.

For the results reported in Section 5 we have used $\kappa = 10$. This value was chosen (somewhat arbitrarily) based on a preliminary analysis, which suggested that the behavior of the algorithm is rather insensitive to the particular value of this parameter. In fact, most of the κ candidate models examined in Step 3b were identical (in our experiments there were typically $1 - 2$ unique models out of the $\kappa = 10$ models that were generated).

As a last remark we note that one may also consider other ways of determining the state-space of the latent variables. One immediate approach is to search for a suitable state-space by fixing the number of states, and use some learning algorithm, see e.g., Dempster et al. (1977), Binder et al. (1997), or Wettig et al. (2003). This can be done greedily to maximize some performance criteria, like BIC, MDL or MDL_p. However, to reduce the computational complexity of the algorithm we have not considered this any further.

4.2. Inference in the learned model

The algorithm for collapsing the state-space of a latent variable is the source of the semantics for these nodes, and in turn the reason why we can represent the HNB as a Naïve Bayes model with aggregations in place of the attributes. This compact representation requires a "deterministic inference engine" to calculate $P(C \mid \boldsymbol{a})$, because the aggregations defined by the semantics of the latent variables can in general not be encoded by the conditional probability tables for the variables. Assume, for instance, that we have three binary variables L, X, Y, $\text{ch}(L) = \{X, Y\}$, and "$L = 1$ if and only if $X = Y$". This relationship cannot be encoded in the model $X \leftarrow L \rightarrow Y$, and to infer the state of the latent variable L from X and Y we would therefore need to design a special inference algorithm which explicitly uses the semantics of L. To alleviate this potential drawback we can simply re-define the network-structure s.t. standard Bayesian inference algorithms (Lauritzen and Spiegelhalter, 1988; Shafer and Shenoy, 1990; Madsen and Jensen, 1998) can be used: Introduce a new latent variable L', and change the network structure s.t. $\text{ch}(L) = \text{pa}(X) = \text{pa}(Y) = \{L'\}$; L' is equipped with at most one state for each possible combination of its children's states. This enlarged structure is capable of encoding any relation between $\{X, Y\}$ and L using only the conditional probability tables specified in the network. Hence, the enlarged structure can be handled by any

standard BN propagation algorithm, and since the structure is still an HNB (although not a parsimonious one) the inference can be performed extremely fast.

4.3. Time complexity of the learning algorithm

The complexity can be analyzed by considering the three steps that characterize the algorithm:

1. Find a candidate latent variable.
2. Find the state-space of the candidate latent variable, and check if it is useful.
3. Iterate until no more candidate latent variables are accepted.

Part 1
Proposing a candidate latent variable corresponds to finding the pair $\{X, Y\}$ of variables with the strongest correlation (Equation 3). There are at most $(n^2 - n)/2$ such pairs, where n is the number of attribute variables. By calculating the conditional mutual information for a pair of variables as well as sorting the pairs (for future iterations) according to this measure we get the time complexity: $O(n^2 \cdot (N + \log(n)))$ In the remainder of this analysis we shall consider $\log(n)$ to be negligible compared to N hence, for Part 1 we get $O(n^2 \cdot (N + \log(n))) \approx O(n^2 \cdot N)$.

Part 2
The time complexity of calculating the gain, $\Delta_L(\cdot, \cdot)$, of collapsing two states is simply $O(N)$, see Equation 5. Due to local decomposability, the gain of collapsing two states has no effect on collapsing two other states, and there are therefore at most $(|\mathrm{sp}(L)|^2 - |\mathrm{sp}(L)|)/2$ such possible combinations to calculate initially. Next, when two states are collapsed, $\Delta_L(\cdot, \cdot)$ must be calculated for $|\mathrm{sp}(L)| - 1$ new state combinations, and the collapsing is performed at most $|\mathrm{sp}(L)| - 1$ times. The time complexity of finding the state-space of a candidate latent variable is therefore $O\left(N \cdot |\mathrm{sp}(L)|^2 + N \cdot |\mathrm{sp}(L)| \left(|\mathrm{sp}(L)| - 1\right)/2\right) = O(|\mathrm{sp}(L)|^2 \cdot N)$.

Having found the cardinality of a candidate variable, say L, we test whether it should be included in the model using the wrapper approach. From the rule-based propagation method it is easy to see that the time complexity of this task is $O(n \cdot N)$. Thus, the time complexity of Part 2 is $O((n + |\mathrm{sp}(L)|^2) \cdot N)$.

Part 3
Each time a latent variable is introduced we would in principle need to

perform the above steps again, and the time complexity would therefore be $n-1$ times the time complexities above. However, by exploiting locality some of the previous calculations can be reused.

Moreover, note that after having introduced a latent variable L with children X and Y, we cannot create another latent variable having either X or Y as a child (due to the structure of the HNB model). Thus, after having included a latent variable the cardinality of the resulting set of candidate pairs is reduced by $n-1$. This implies that we will perform at most $n-2$ re-initializations, and the overall time complexity of the algorithm is therefore $O(n^2 \cdot N + n \cdot (n \cdot N + (|\mathrm{sp}\,(L)|^2 \cdot N))) = O(n^2 \cdot |\mathrm{sp}\,(L)|^2 \cdot N)$.

It is important to emphasize that the computational complexity is dependent on the cardinality of the latent variables. In the worst case, if none of the states are collapsed, then the cardinality of a latent variable L is exponential in the number of leaves in the subtree having L as root. However, this situation also implies that the leaves are conditionally dependent given the class variable, and that the leaves and the class variable do not exhibit any type of context-specific independence (Boutilier et al., 1996).

The time complexity observed in practice during the empirical study is summarized in Appendix A.

5. Empirical results

In this section we will investigate the merits of the proposed learning algorithm by using it to learn classifiers for a number of different datasets taken from the Irvine Machine Learning Repository (Blake and Merz, 1998). The datasets were selected based on their previous usage in similar types of analysis, e.g., Friedman et al. (1997) and Grossman and Domingos (2004); see Table I for a summary of the 22 datasets used in this empirical study.

5.1. Accuracy results

We have compared the results of the HNB classifier to those of the Naïve Bayes model (Duda and Hart, 1973), the TAN model (Friedman et al., 1997), the original HNB algorithm (Zhang et al., 2003), C5.0 (Quinlan, 1998), a standard implementation of logistic regression, and neural networks with one hidden layer trained by back-propagation.[10] As some of the learning algorithms (including Algorithm 3) require discrete variables, the attributes were discretized using the entropy-based method of Fayyad and Irani (1993). In addition, instances containing missing

Table I. Datasets used in the experiments

Dataset	#Att	#Cls	Size	Database	#Att	#Cls	Size
postop	8	3	90	cleve	13	2	296
iris	4	3	150	wine	13	3	178
monks-1	6	2	432	thyroid	5	3	215
car	6	4	1728	ecoli	7	8	336
monks-3	6	2	432	breast	10	2	683
glass	9	7	214	vote	16	2	435
glass2	9	2	163	crx	15	2	653
diabetes	8	2	768	australian	14	2	690
heart	13	2	270	chess	36	2	3199
hepatitis	19	2	155	vehicle	18	4	846
pima	8	2	768	soybean-large	35	19	562

A summary of the 22 databases used in the experiments: #Att indicates the number of attributes; #Cls is the number of classes; Size is the number of instances. 5-fold cross validation was used for all datasets. Further details regarding the datasets can be found at the UCI Machine Learning Repository.

attribute-values were removed; all pre-processing was performed using MLC++ (Kohavi et al., 1994).

The accuracy-results for the data sets are given in Appendix B (Table IV) and a summary can be seen in Table II. In this table we report the number of datasets for which each classifier is the best overall (#Winner). Note that the sum of these numbers is larger than the number of datasets because we have several ties. For a specific classifier, #Draw is the number of datasets where the classifier is not the best overall, but is not significantly worse than the winner either (at the 10% level). The last three columns give the number of datasets for which the classifier is significantly poorer than the winner at 10%, 5%, and 1% level, respectively (see Appendix B for more information). In particular, we note that the proposed HNB classifier achieves the best result, among all the classification algorithms, for 12 of the 22 datasets, and draws with the winner for 8 of the other ones. The HNB algorithm is significantly poorer than the winner (at 10% level) for only 2 of the 22 datasets. Finally, Figure 2 gives a graphical illustration of the accuracy results of the proposed algorithm compared to the accuracy results of the algorithms mentioned above.

To help the interpretation of the results reported in Table II and Table IV we ran all algorithms with their "basic" configurations, i.e., no

optimization was performed. One notable exception is the TAN model, which was run both in its basic form and with a damping-factor equal to 5 virtual counts (this algorithm is denoted TAN-5 in Table II and Table IV). The computational complexity of the algorithm by Zhang et al. (2003) prevented us from obtaining results for this algorithm from the three most complex domains.

Table II. Summary of classification results

Algorithm	#Winner	#Draw	$\#p < 10\%$	$\#p < 5\%$	$\#p < 1\%$
NB	6	10	6	5	3
TAN	2	12	8	5	1
TAN-5	5	10	7	3	1
C5.0	4	10	8	6	1
NN	0	11	11	8	0
Logistic	2	11	9	4	2
Zhang et al.	1	5	13	12	7
HNB	12	8	2	1	1

5.2. Learning curve

We have analyzed the effect that the size of training data has on the classification accuracy of our learning algorithm; we shall use $\alpha(N)$ to denote the accuracy of a learning algorithm when trained on a database of size N. The relation between N and $\alpha(N)$ (called the *learning curve*) can provide important insight into the workings of the algorithms. For example, it is well known that the Naïve Bayes algorithm learns fast (relatively "high" accuracy for "small" databases), but because of the strong learning bias it also has a tendency to converge too quickly s.t. even for "large" databases, the accuracy does not increase beyond a certain level, see also Ng and Jordan (2002).

To estimate the learning curve (of a given algorithm) for a specific data set we first made a random sub-sampling (without replacement) of N cases from that data set. These cases were then used as training data for the algorithm in question, and the remaining cases were used as test data; this was repeated 10 times for each N. As an example, consider Figure 3, which shows the learning curves for the car database and the postop database.

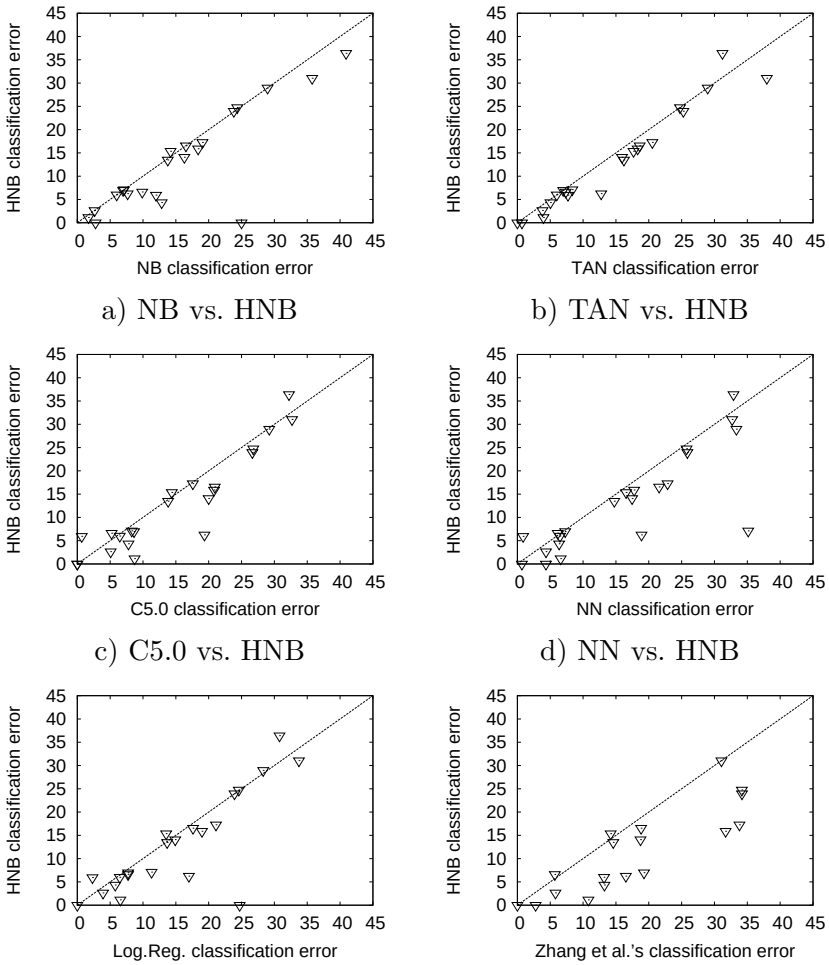
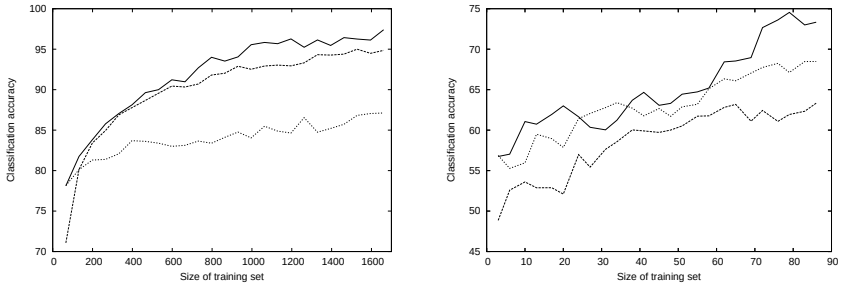


Figure 2. Scatter plot of classification error for HNB and a selection of other classification systems. In each plot, a point represents a dataset. The HNB's classification error is given on the y-axis, whereas the other system's error is given on the x-axis. Hence, data points below the diagonal corresponds to datasets where the HNB is superior, whereas points above the diagonal are datasets where the HNB classifier is inferior to the other system.

For the results shown in Figure 3, we note that the proposed learning algorithm appears to generate classifiers with the same fast starting property as the Naïve Bayes classifier, but, at the same time, avoids the premature convergence problem of these classifiers.



(a) car (b) postop

Figure 3. The figures show the estimated learning curves for Algorithm 3 (solid line), TAN (dashed line) and Naïve Bayes (dotted line). Part (a) gives the results for the car database, and the results for the postop database are shown in part (b).

To examine this further, we generated learning curves for *all* the datasets used in the experiments. To be able to compare the learning curves from, e.g., postop (with only 90 cases) to the results of chess (with 3199 cases), we first scaled the values on the x-axis to give the percentage of the full dataset that was used as the training set.[11] Next, in order to compare the accuracy results for the different datasets we normalized the results s.t. the best accuracy result obtained (for all data sizes and all classification algorithms) would correspond to 100 "points", whereas the worst result was given a score of 0 "points". Finally, we calculated the score obtained on each dataset when x percent of the available data was used. The averages of these values are shown on the y-axis in Figure 4 (a and b) as a function of the percentage of the data used as training data; the empirical standard deviations are shown in Figure 4(b) using error bars.

Finally, Figure 5 shows the average number of latent variables inserted by the learning algorithm as a function of the size of the training set. The results are from the car database, and are averaged over 10 runs. We can see that the algorithm has a built-in ability to avoid overfitting when working with this data-set; only a few latent variables are inserted when the size of the training set is small, and the algorithm seems to stabilize around 3 latent variables for $N \geq 1000$ training cases.

5.3. Semantic interpretation of latent structures

In some domains the learned HNB models may contain a latent structure, which is amenable to interpretation. In this subsection we will consider the car database, a synthetic data-set consisting of 1728 cases which describe the relationship between certain characteristics of a car

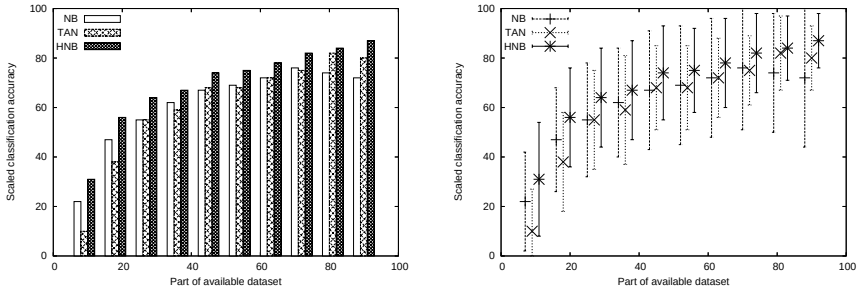


Figure 4. The figures show the scaled learning curves for the proposed HNB algorithm (Algorithm 3), TAN, and Naïve Bayes. The x-axis show the percentage of the database used to obtain the results, and on the y-axis we report the average scaled efficiency. Figure (a) shows a histogram representation (included for ease of interpretation) of the average results, whereas figure (b) also shows the empirical standard deviation using error-bars (an error-bar indicates 1 standard deviation of the estimate).

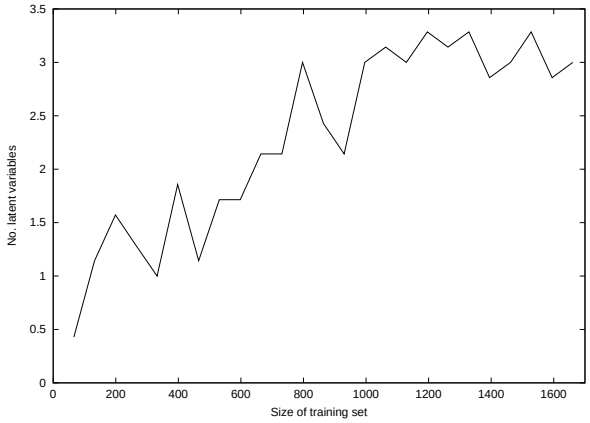


Figure 5. The number of latent variables inserted by the proposed HNB algorithm (Algorithm 3), as a function of the size of the training set (the numbers are from the car domain).

and whether or not the car is "acceptable". We have chosen the car database due to its intuitive interpretation and because its specification also covers the gold standard model from which the data was generated. The states of the class attribute are unacceptable, acceptable, good and very-good, and the attributes describing the car are: number of doors (`Doors`), capacity in term of persons (`Persons`), size of luggage boot (`Lug_boot`), estimated safety of the car (`Safety`), buying

price (Buying), and price of maintenance (Maint). When applying the algorithm to this database we obtain the model illustrated in Figure 6.

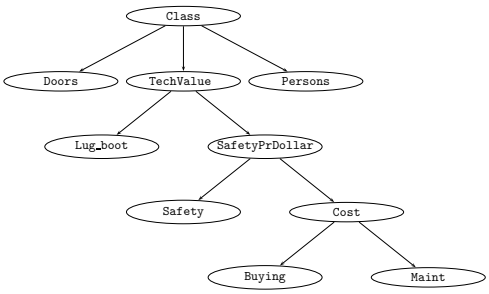


Figure 6. The learned HNB model for the car database with three latent variables.

The nodes TechValue, SafetyPrDollar and Cost correspond to the latent variables identified by the algorithm; the names of these variables have manually been deduced from their usage in the model. For example, the node Cost summarizes the two types of monetary costs using the four states Cost=very-high, Cost=high, Cost=medium and Cost=low, where e.g. the state very-high encodes the following configurations of the attributes Maint and Buying:

$$[(\texttt{Maint} = \text{very-high}) \wedge (\texttt{Buying} = \text{very-high})]$$
$$\vee$$
$$[(\texttt{Maint} = \text{very-high}) \wedge (\texttt{Buying} = \text{high})]$$
$$\vee$$
$$[(\texttt{Maint} = \text{high}) \wedge (\texttt{Buying} = \text{very-high})].$$

The node SafetyPrDollar compares the safety level to the cost, and contains the six states very-high, high, good, medium, low and very-low. For instance, SafetyPrDollar=very-low specify [Safety = low $\vee$ Cost = very-high].

We conclude this subsection by reiterating that Algorithm 3 has the sole purpose of generating HNB models that achieve high classification accuracy. The semantic interpretation presented above is therefore a spin-off of the model definition rather than a required feature of our learning algorithm. Hence, we do not claim that all HNB models will have interesting latent structures; rather we suggest that it *may* be worthwhile to examine an HNB's latent structure. In some cases, this will give a decision maker insight into the rules which govern the classification of some instance, and may thereby increase the user's confidence in the model.

6. Concluding remarks and future work

In this paper we have proposed an algorithm for learning hierarchical Naïve Bayes models for classification. Experimental results have shown that the learned classifiers offer results that are significantly better than those of other commonly used classification methods. Moreover, a number of existing tools may be able to improve the classification accuracy even further, e.g., supervised learning of the probability parameters (Wettig et al., 2003).

As part of future research, we recognize that Step 3a of Algorithm 3 may require further investigation. In this step we pick out κ candidate models to represent the search boundary, and then select a single model from this set. We plan to investigate the way the candidates are selected. In particular, the structure of a candidate model is determined by a conditional independence test, which is not necessarily related to the classification accuracy. Other methods may be considered (e.g., exhaustive search), and they should be compared to the effectiveness of the proposed heuristic. In an initial investigation, where we compared our results with the results of performing an exhaustive search, the heuristic showed good performance, but this should be examined more closely before any conclusions can be drawn. The same point can be made regarding the way state-spaces are selected (Algorithm 2): Other methods can be envisioned, including exhaustive search as well as search algorithms based on greedily maximizing some score function for a fixed number of states.

Acknowledgements

We have benefited from interesting discussions with the members of the Machine Intelligence group at Aalborg University, in particular Tomás Kočka, and Jiří Vomlel. We thank Nevin L. Zhang for his valuable insight and for giving us access to his implementation of the HNB learning algorithm by Zhang et al. (2003), and the anonymous reviewers for very helpful comments on an earlier version of the paper. Finally, we would like to thank *Hugin Expert* (`http://www.hugin.com/`) for giving us access to the *Hugin Decision Engine*, which forms the basis of our implementation.

Notes

[1] We will not consider regular HNB models with singly connected latent variables.

[2] Note that Zhang et al. (2003) do not consider whether a regular model is parsimonious or not. When we later search the space of all regular models looking for a "good" classifier we may therefore not use the smallest search space that define all parsimonious HNB classifiers.

[3] Since HNBs are only defined for discrete variables, all continuous variables should be discretized before the learning algorithm is deployed.

[4] Note that restricting the latent variables to only having two children does not affect the expressibility of the models.

[5] Note that the search procedure ensures that we will at most make $n-1$ model selections (step c).

[6] One might also consider other measures for testing for conditional dependences.

[7] The relevant subtree can also be seen as the part of the classifier structure that is directly affected by the potential collapse of the states l_i and l_j.

[8] The problem with irregular HNB models appears when there exists a variable $X \in \mathrm{ch}(L)$ s.t. $|\mathrm{sp}(X)| > |\mathrm{sp}(C)|$.

[9] Note that the semantics also allow the decision maker to inspect the "rules" that form the basis of a given classification. Through this insight she can e.g. consider whether the classification of the system should be overruled or accepted.

[10] We used Clementine (SPSS Inc., 2002) to generate the C5.0, logistic regression and neural network models.

[11] The result obtained using, e.g., 60% of the postop database as training data should be compared to the result obtained using 60% of chess database as training data.

References

Binder, J., D. Koller, S. Russell, and K. Kanazawa: 1997, 'Adaptive probabilistic networks with hidden variables'. *Machine Learning* **29**(2–3), 213–244.

Blake, C. and C. Merz: 1998, 'UCI repository of machine learning databases'. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Boutilier, C., N. Friedman, M. Goldszmidt, and D. Koller: 1996, 'Context-specific independence in Bayesian networks'. In: *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*. San Fransisco, CA., pp. 115–123.

Chow, C. K. and C. Liu: 1968, 'Approximating discrete probability distributions with dependence trees'. *IEEE Transactions on Information Theory* **14**, 462–467.

Dempster, A. P., N. M. Laird, and D. B. Rubin: 1977, 'Maximum likelihood from incomplete data via the EM algorithm'. *Journal of the Royal Statistical Society, Series B* **39**, 1–38.

Domingos, P. and M. Pazzani: 1997, 'On the optimality of the simple Bayesian classifier under zero-one loss'. *Machine Learning* **29**(2–3), 103–130.

Duda, R. O. and P. E. Hart: 1973, *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Elidan, G. and N. Friedman: 2001, 'Learning the dimensionality of hidden variables'. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA., pp. 144–151, Morgan Kaufmann Publishers.

Fayyad, U. M. and K. B. Irani: 1993, 'Multi-interval discretization of continuous-valued attributes for classification learning.'. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA., pp. 1022–1027, Morgan Kaufmann Publishers.

Friedman, J. H.: 1997, 'On bias, variance, 0/1-loss, and the curse of dimensionality'. *Data Mining and Knowledge Discovery* **1**(1), 55–77.

Friedman, N., D. Geiger, and M. Goldszmidt: 1997, 'Bayesian network classifiers'. *Machine Learning* **29**(2–3), 131–163.

Greiner, R., A. J. Grove, and D. Schuurmans: 1997, 'Learning Bayesian nets that perform well'. In: *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. San Fransisco, CA., pp. 198–207, Morgan Kaufmann Publishers.

Grossman, D. and P. Domingos: 2004, 'Learning Bayesian network classifiers by maximizing conditional likelihood'. In: *Proceedings of the Twentyfirst International Conference on Machine Learning*. Banff, Canada, pp. 361–368, ACM Press.

Jaeger, M.: 2003, 'Probabilistic classifiers and the concepts they recognize'. In: *Proceedings of the Twentieth International Conference on Machine Learning*. Menlo Park, pp. 266–273, The AAAI Press.

Jensen, F. V.: 2001, *Bayesian Networks and Decision Graphs*. New York, NY: Springer-Verlag.

Kohavi, R.: 1995, 'A study of cross-validation and bootstrap for accuracy estimation and model selection'. In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA., pp. 1137–1143, Morgan Kaufmann Publishers.

Kohavi, R., G. John, R. Long, D. Manley, and K. Pfleger: 1994, 'MLC++: A machine learning library in C++'. In: *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*. pp. 740–743, IEEE Computer Society Press.

Kohavi, R. and G. H. John: 1997, 'Wrappers for feature subset selection'. *Artificial Intelligence* **97**(1–2), 273–324.

Kononenko, I.: 1991, 'Semi-naive Bayesian classifier'. In: *Proceedings of Sixth European Working Session on Learning*. Porto, Portugal, pp. 206–219, Springer-Verlag.

Kočka, T. and N. L. Zhang: 2002, 'Dimension correction for hierarchical latent class models'. In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA., pp. 267–274, Morgan Kaufmann Publishers.

Lam, W. and F. Bacchus: 1994, 'Learning Bayesian belief networks: An approach based on the MDL principle'. *Computational Intelligence* **10**(4), 269–293.

Langley, P.: 1993, 'Induction of recursive Bayesian classifiers'. In: *Proceedings of the Fourth European Conference on Machine Learning*, Vol. 667 of *Lecture Notes in Artificial Intelligence*. pp. 153–164, Springer-Verlag.

Langley, P.: 1994, 'Selection of relevant features in machine learning'. In: *Proceedings of the AAAI Fall symposium on Relevance*. The AAAI Press.

Lauritzen, S. L. and D. J. Spiegelhalter: 1988, 'Local computations with probabilities on graphical structures and their application to expert systems'. *Journal of the Royal Statistical Society, Series B* **50**(2), 157–224.

Madsen, A. L. and F. V. Jensen: 1998, 'Lazy propagation in junction trees'. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. pp. 362–369, Morgan Kaufmann Publishers.

Martin, J. D. and K. VanLehn: 1994, 'Discrete factor analysis: Learning hidden variables in Bayesian networks'. Technical Report LRDC-ONR-94-1, Department of Computer Science, University of Pittsburgh. `http://www.pitt.edu/~vanlehn/distrib/Papers/Martin.pdf`.

Mitchell, T. M.: 1997, *Machine Learning*. Boston, MA.: McGraw Hill.

Nadeau, C. and Y. Bengio: 2003, 'Inference for the generalization error'. *Machine Learning* **52**(3), 239–281.

Ng, A. Y. and M. I. Jordan: 2002, 'On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes'. In: *Advances in Neural Information Processing Systems 15*. Vancouver, British Columbia, Canada, pp. 841–848, The MIT Press.

Pazzani, M.: 1996a, 'Searching for dependencies in Bayesian classifiers'. In: *Learning from data: Artificial Intelligence and Statistics V*. New York, N.Y., pp. 239–248.

Pazzani, M. J.: 1996b, 'Constructive induction of Cartesian product attributes'. In: *ISIS: Information, Statistics and Induction in Science*. Singapore, pp. 66–77, World Scientific.

Pearl, J.: 1988, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA.: Morgan Kaufmann Publishers.

Quinlan, R.: 1998, 'C5.0: An informal tutorial'. `http://www.rulequest.com/see5-unix.html`.

Rissanen, J.: 1978, 'Modelling by shortest data description'. *Automatica* **14**, 465–471.

Schwarz, G.: 1978, 'Estimating the dimension of a model'. *The Annals of Statistics* **6**, 461–464.

Shafer, G. R. and P. P. Shenoy: 1990, 'Probability propagation'. *Annals of Mathematics and Artificial Intelligence* **2**, 327–352.

Spirtes, P., C. Glymour, and R. Scheines: 1993, *Causation, Prediction, and Search*. New York: Springer-Verlag.

SPSS Inc.: 2002, 'Clementine v6.5'. `http://www.spss.com/spssbi/clementine/`.

Wettig, H., P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri: 2003, 'When discriminative learning of Bayesian network parameters is easy'. In: *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. pp. 491–496, Morgan Kaufmann Publishers.

Whittaker, J.: 1990, *Graphical models in applied multivariate statistics*. Chichester, UK: John Wiley & Sons.

Zhang, H.: 2004a, 'The optimality of naive Bayes'. In: *Proceedings of the Seventeenth Florida Artificial Intelligence Research Society Conference*. pp. 562–567, The AAAI Press.

Zhang, N. L.: 2004b, 'Hierarchical latent class models for cluster analysis'. *Journal of Machine Learning Research* **5**(6), 697–723.

Zhang, N. L., T. D. Nielsen, and F. V. Jensen: 2003, 'Latent variable discovery in classification models'. *Artificial Intelligence in Medicine* **30**(3), 283–299.

Appendix

A. Empirical complexity results

In practice, the main computational effort of our algorithm goes into calculating the conditional mutual information (CMI) between all pairs of variables given the class variable (Step 3a of Algorithm 3). To give an indication of the (relative) run time complexity of the algorithm, we report on the number of times the algorithm calculates CMI ("CMI; HNB" in Table III). These numbers are easiest to interpret relative to the TAN algorithm (Friedman et al., 1997) that also relies on CMI

calculations ("CMI; TAN" in Table III), i.e., the TAN algorithm may be considered a baseline used for comparison. Finally, the ratio between these two numbers are listed as the "CMI ratio". The CMI ratio is bounded below by κ, the number of candidate models used to represent the search boundary (recall that $\kappa = 10$ in our experiments), and is also roughly linear in this value. For ease of reference, we have included information regarding the number of attributes ("#Att") of each dataset. We also give the number of latent variables inserted by the algorithm ("#Latent"), both as a range as well as a mean plus standard deviation.

In our experiments, Algorithm 3 has running times that are between 6 and 25 times those of the TAN algorithm when $\kappa = 10$ is kept fixed; the running times include both learning and inference w.r.t. the test dataset. In particular, a ratio below 10 occurs when the average number of latent variables inserted by the HNB algorithm is close to zero, i.e., when we are working with an NB model where the time complexity for inference is smaller than that of a TAN model. It should be noted that these run-time ratios should not to be taken literally, but rather be seen as indications of the run-time complexity: our implementation of the HNB algorithm is far from optimized and is implemented with focus on debugging capabilities.

B. Accuracy results

The detailed accuracy-results for all the data sets are shown in Table IV. For each dataset we have estimated the accuracy of each classifier (in percentage of instances which are correctly classified). We note that the true ability of each classifier can only be calculated "correctly" if we know the probability distribution $P(C, \boldsymbol{A})$. As we only have datasets of limited sizes available, the estimated accuracies are random variables (the uncertainty stems from the fact that a dataset is only a sample from this unknown distribution). We therefore give a standard deviation of the accuracy estimate. The numbers we present are the theoretical values calculated according to Kohavi (1995), and are not necessarily the same as the empirical standard deviations observed during cross validation.

The uncertainty in the estimated accuracies forces us to use a statistical test to decide if one classifier is better than another on a particular domain. We use Nadeau and Bengio (2003)'s *corrected resampled t-test*. This test takes the calculated accuracy on each cross-validation fold into consideration, and thereby tries to minimize the uncertainty of the estimate outlined above; the same cross-validation folds were given

Table III. Number of times the HNB and TAN algorithms calculate conditional mutual information for given datasets; $\kappa = 10$

Database	#Att	#Latent		CMI; HNB	CMI; TAN	CMI ratio
postop	8	$[0;1]$	0.2 ± 0.45	1100	105	10.5
iris	4	$[0;1]$	0.2 ± 0.45	320	30	10.7
monks-1	6	$[2;2]$	2.0 ± 0.00	950	75	12.7
car	6	$[2;3]$	2.8 ± 0.45	1180	75	15.7
monks-3	6	$[2;2]$	2.0 ± 0.00	950	75	12.7
glass	9	$[0;0]$	0.0 ± 0.00	1050	105	10.0
glass2	9	$[0;1]$	0.2 ± 0.45	530	50	10.6
diabetes	8	$[0;1]$	0.2 ± 0.45	590	56	10.5
heart	13	$[0;3]$	1.2 ± 1.30	1790	149	12.0
hepatitis	19	$[0;1]$	0.2 ± 0.45	8720	855	10.2
pima	8	$[0;1]$	0.4 ± 0.55	1520	140	10.9
cleve	13	$[0;2]$	0.6 ± 0.55	2920	265	11.0
wine	13	$[0;1]$	0.2 ± 0.45	4010	390	10.3
thyroid	5	$[0;1]$	0.4 ± 0.56	560	50	11.2
ecoli	7	$[0;1]$	0.2 ± 0.45	740	70	10.6
breast	10	$[0;4]$	0.8 ± 1.79	2020	180	11.2
vote	16	$[0;3]$	1.4 ± 1.52	6920	600	11.5
crx	15	$[0;1]$	0.4 ± 0.56	5510	520	10.6
australian	14	$[0;1]$	0.4 ± 0.56	4790	455	10.5
chess	36	$[2;2]$	2.0 ± 0.00	34850	3150	11.1
vehicle	18	$[0;4]$	1.6 ± 1.52	9150	765	12.0
soybean-large	35	$[1;4]$	2.2 ± 1.30	33280	2975	11.2

to all classification algorithms. The best result for each dataset is given in boldface. Results that are significantly poorer than the best on a given dataset at 10%-level are marked with '*'. Results significant at 5%-level are marked with '$^\circ$', and 1%-level with '$^\bullet$'.

Table IV. Classifier accuracies

Database	NB	TAN	TAN-5	C5.0	NN	Logistic	Zhang et al.	HNB
postop	64.25+/-5.0	63.03+/-5.1	*62.09+/-5.1	67.31+/-4.9	67.31+/-4.9	66.26+/-5.0	68.94+/-4.9	**68.95+/-4.9**
iris	**94.00+/-2.0**	**94.00+/-2.0**	**94.00+/-2.0**	93.53+/-2.0	93.51+/-2.0	93.53+/-2.0	°86.81+/-2.8	**94.00+/-2.0**
monks-1	•75.00+/-2.2	**100.0+/-0.1**	**100.0+/-0.1**	**100.0+/-0.1**	*95.65+/-1.0	•75.28+/-2.1	**100.0+/-0.1**	**100.0+/-0.1**
car	•87.15+/-0.8	94.97+/-0.5	*93.86+/-0.6	•92.21+/-0.6	*93.65+/-0.6	*94.23+/-0.6	•86.75+/-0.8	**95.66+/-0.5**
monks-3	°97.22+/-0.8	99.30+/-0.4	*98.84+/-0.5	**100.0+/-0.1**	99.31+/-0.4	**100.0+/-0.1**	°97.22+/-0.8	**100.0+/-0.1**
glass	71.04+/-3.1	71.04+/-3.1	**72.44+/-3.0**	70.78+/-3.1	66.66+/-3.2	71.70+/-3.1	•53.45+/-3.4	71.04+/-3.1
glass2	81.61+/-3.0	81.69+/-3.0	82.29+/-3.0	*79.18+/-3.2	82.23+/-3.0	81.02+/-3.1	•68.30+/-3.6	**84.11+/-2.9**
diabetes	**75.65+/-1.5**	75.25+/-1.6	75.51+/-1.6	73.21+/-1.6	°74.25+/-1.6	75.41+/-1.6	•65.83+/-1.7	75.25+/-1.6
heart	83.70+/-2.2	84.07+/-2.2	84.44+/-2.2	°80.00+/-2.4	°82.54+/-2.3	85.09+/-2.2	°81.26+/-2.4	**85.93+/-2.3**
hepatitis	92.34+/-2.1	87.25+/-2.7	87.27+/-2.7	°80.64+/-3.2	°81.11+/-3.1	°83.01+/-3.0	*83.49+/-3.0	**93.76+/-2.1**
pima	**76.17+/-1.5**	°74.74+/-1.6	°74.87+/-1.6	°73.35+/-1.6	°74.13+/-1.6	76.07+/-1.5	•65.81+/-1.7	76.04+/-1.5
cleve	**83.46+/-2.1**	81.38+/-2.2	82.41+/-2.2	*79.07+/-2.4	*78.42+/-2.4	82.38+/-2.2	81.17+/-2.3	83.45+/-2.1
wine	98.30+/-1.0	°96.03+/-1.5	°96.03+/-1.5	°91.28+/-2.1	°93.41+/-1.9	°93.44+/-1.9	°89.19+/-2.3	**98.86+/-0.8**
thyroid	93.02+/-1.7	93.02+/-1.7	**94.42+/-1.6**	91.36+/-1.9	92.73+/-1.8	92.27+/-1.8	•80.73+/-2.7	93.02+/-1.7
ecoli	80.95+/-2.1	*79.76+/-2.2	*80.06+/-2.2	82.41+/-2.1	°77.14+/-2.3	*78.89+/-2.2	•66.20+/-2.6	**82.74+/-2.1**
breast	**97.36+/-0.6**	*96.19+/-0.7	96.78+/-0.7	°94.92+/-0.8	°95.64+/-0.8	*96.08+/-0.7	°94.20+/-0.9	**97.36+/-0.6**
vote	*90.11+/-1.4	*92.64+/-1.3	94.48+/-1.1	**94.77+/-1.1**	93.86+/-1.2	*92.27+/-1.3	94.30 +/- 1.1	93.39+/-1.3
crx	86.22+/-1.3	°83.78+/-1.4	85.30+/-1.3	86.17+/-1.4	85.25+/-1.4	86.33+/-1.3	85.41 +/- 1.4	**86.51+/-1.3**
australian	85.80+/-1.3	°82.32+/-1.5	84.78+/-1.4	85.61+/-1.3	83.88+/-1.4	**86.47+/-1.3**	85.80 +/- 1.3	84.64+/-1.4
chess	•88.02+/-0.6	•92.30+/-0.5	•92.30+/-0.5	**99.32+/-0.1**	99.13+/-0.2	•97.69+/-0.3	—	•94.06+/-0.4
vehicle	°59.09+/-1.7	68.79+/-1.6	**69.50+/-1.6**	67.78+/-1.6	67.10+/-1.6	69.22+/-1.6	—	*63.59+/-1.7
soybean-large	**92.90+/-1.0**	91.64+/-1.1	92.17+/-1.0	91.71+/-1.2	°64.86+/-2.0	*88.69+/-1.3	—	92.89+/-1.1

Classifiers accuracies: Calculated accuracy for the 22 datasets used in the experiments; the results are given together with their theoretical standard deviation. The adjusted t-test of Nadeau and Bengio (2003) was used to compare the classifiers: Results that are significantly poorer than the best on a given dataset at 10%-level are marked with '*'. Results significant at 5%-level are marked with '°', and 1%-level with '•'.